# 機械学習のための
# データ前処理の基本と実践法
# 7

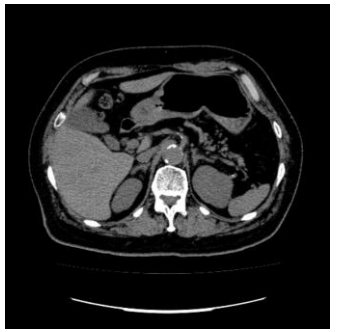
画像認識、画像特徴量、畳み込みニューラルネットワーク

徳島大学
デザイン型AI教育研究センター
**鳥井 浩平**

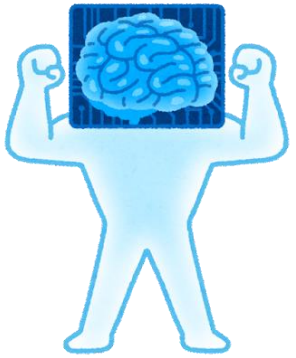
# 自己紹介

名前：**鳥井 浩平**（とりい こうへい）
所属：デザイン型AI教育研究センター
職位：特任助教
趣味：ゲーム、小説（ミステリー・サスペンス）、ドライブ

- 去年まで徳島大学に学生として在籍
- 医用画像に対するAI技術の応用について研究
- 材料化学×AIにも興味あり



✖



# 論文

鳥井 浩平、誉田 栄一、北 研二　doi:10.11242/dentalradiology.62.24

AIを用いたパノラマX線画像からのカルテ入力支援システムの開発

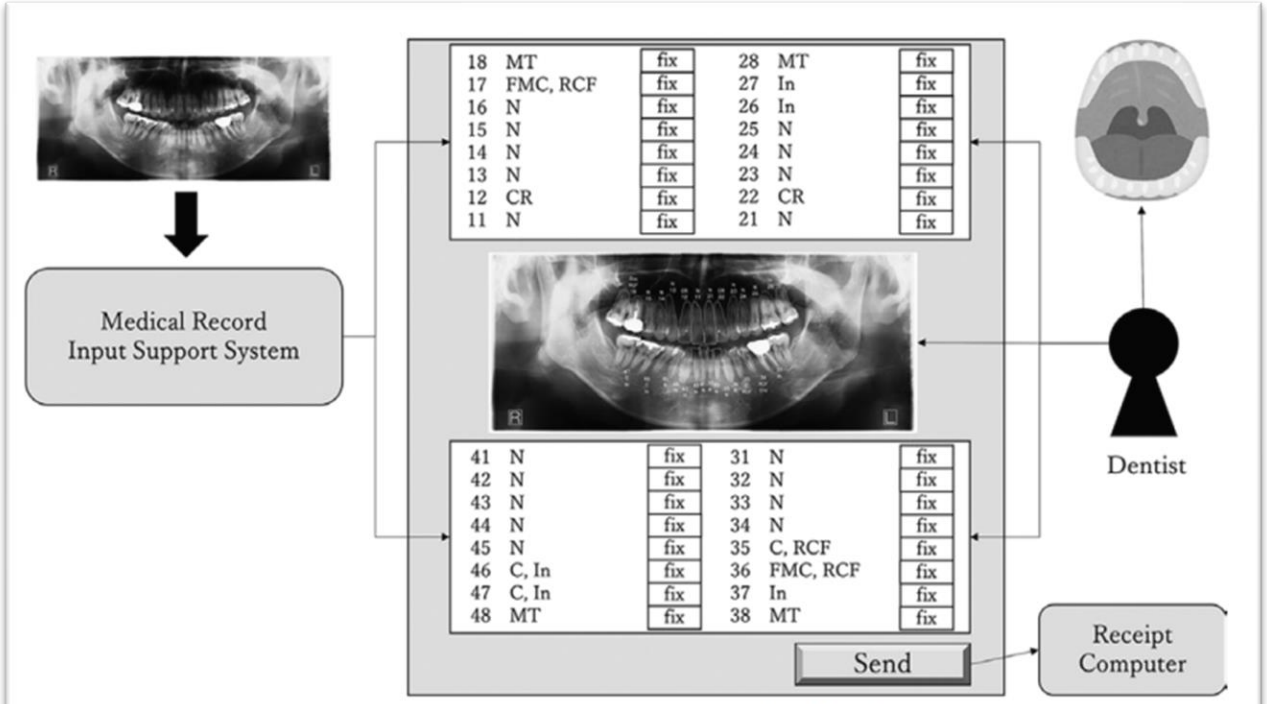



**Figure 1**　本システムによるカルテ入力支援の概略図
歯科医は視診と読影を行い，カルテ入力支援システムが出力したカルテの情報を修正する。

# 今回の内容

1. 画像認識
2. 画像特徴量
3. 畳み込みニューラルネットワーク
   - 今回は概論のみ述べる
   - 次回、実装と併せて細かいところを見ていく

# 7.1 画像認識

画像認識についておさらい

# 画像認識とは

- 画像に何が写っているのかを認識すること
- 深層学習モデルの導入により精度が飛躍的に向上した

入力

**画像認識システム**

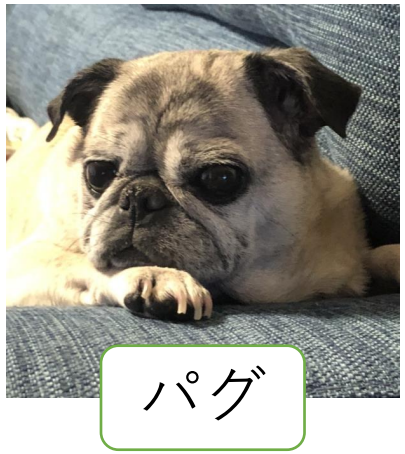
出力

パグ

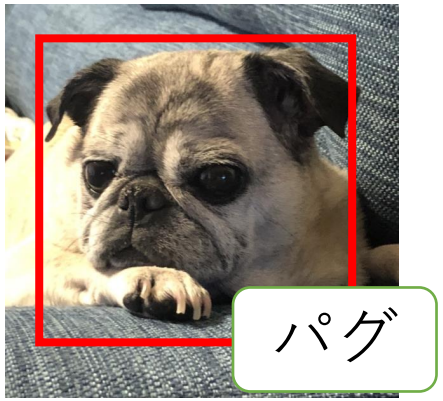
# 画像認識の種類

- 画像認識は主に以下の3つに分けられる
  **画像分類**：画像全体に対して種類を予測する
  **物体検出**：物体の位置と物体の種類を予測する
  **セグメンテーション**：画素単位で種類を予測する
- 画像分類＜物体検出＜セグメンテーションの順に難易度が上がる

画像分類




物体検出




セグメンテーション

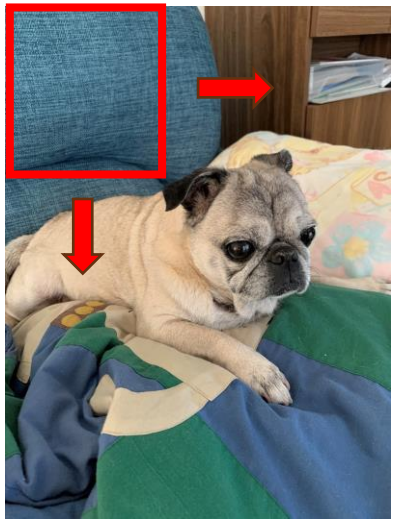
# 従来の画像認識

- 深層学習が登場するまではどのように画像認識を行っていたのか？
- 例として以下をピックアップする

1. テンプレートマッチング
2. 画像検索
3. 画像分類

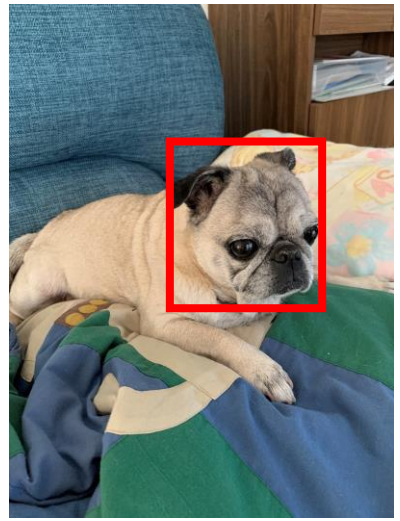
# テンプレートマッチング

1. 探したい物体の画像（テンプレート画像）を用意する
2. 入力画像の一部とテンプレート画像の**類似度**を計算する
3. 入力画像のすべての領域に対して2を行う
4. 最も類似度が高い入力画像の一部を検出結果とする



入力画像

テンプレート画像

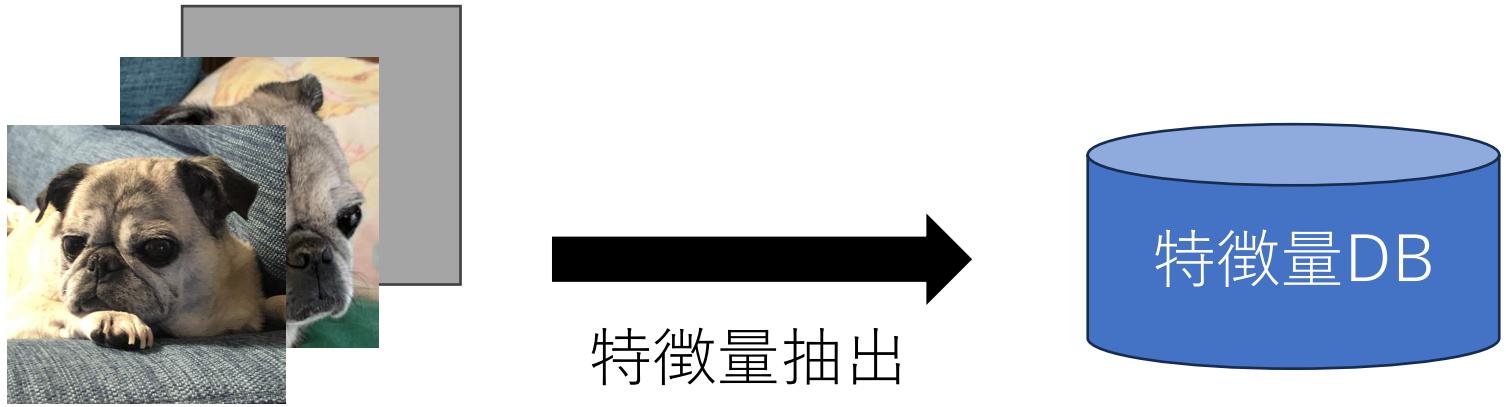


検出結果

# 画像検索

登録処理

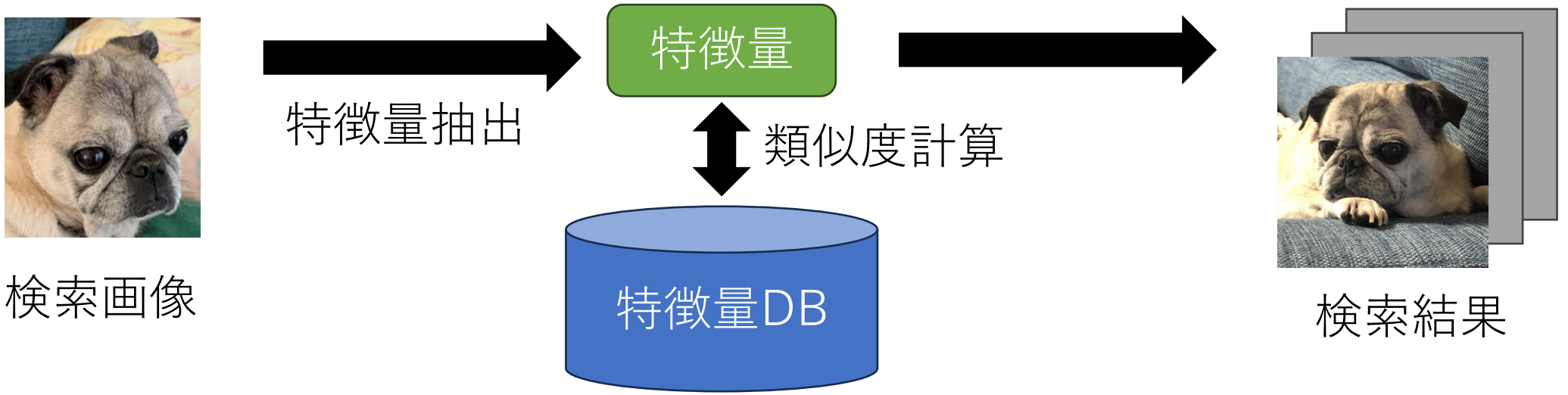
1. 大量の画像を収集する
2. 各画像の**特徴量（多次元ベクトル）**を抽出する
3. 抽出した特徴量を特徴量データベースに保存する



# 画像検索

検索処理

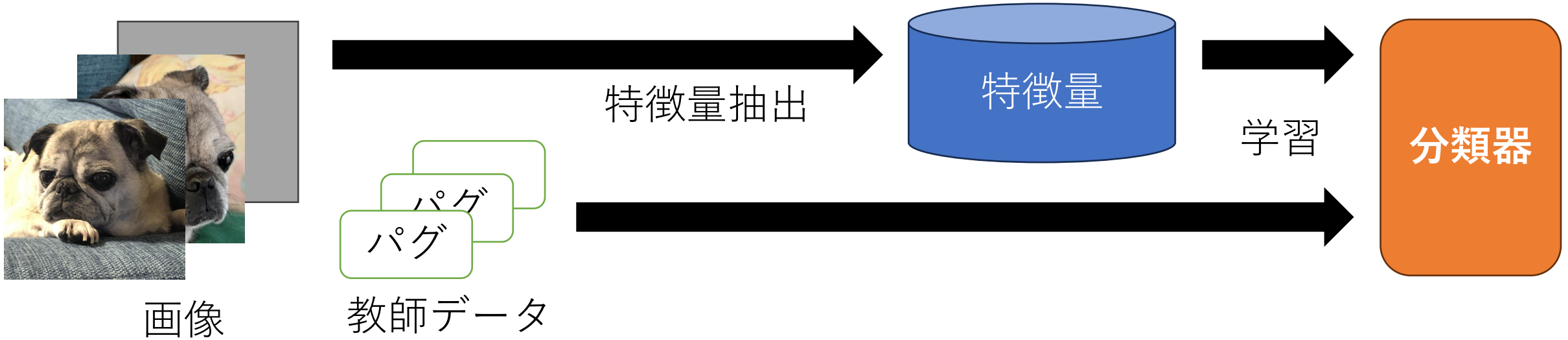
1. 検索したい画像を用意する
2. 特徴量を抽出する
3. 抽出した特徴量と特徴量データベースのデータとの類似度を計算する



# 画像分類

学習フェーズ

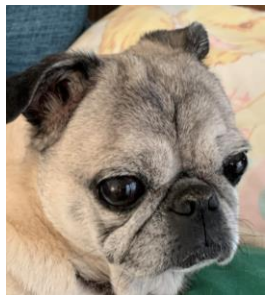
1. 大量の画像を収集し、各画像に対して**教師データ**を作成する
2. 各画像の特徴量を抽出する
3. 抽出した特徴量と教師データを用いて**分類器**の学習を行う
   分類器の例）SVM、ランダムフォレストなど



# 画像分類

分類フェーズ

1. 分類したい画像を用意する
2. 特徴量を抽出する
3. 抽出した特徴量を学習済み分類器に入力して結果を得る



分類したい画像

特徴量抽出

特徴量

入力

学習済み
分類器

出力

パグ

# （補足）類似度の例

類似度≒2つのデータ間の距離

ユークリッド距離

- 2つのベクトルがどれくらい離れているかを表す

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

コサイン距離

- 2つのベクトルがどれくらい違う方向を向いているかを表す

$$\cos(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} {x_i}^2}\sqrt{\sum_{i=0}^{n} {y_i}^2}}$$

# 7.2 画像特徴量

さまざまな画像特徴量を見てみよう

# 画像特徴量の推移

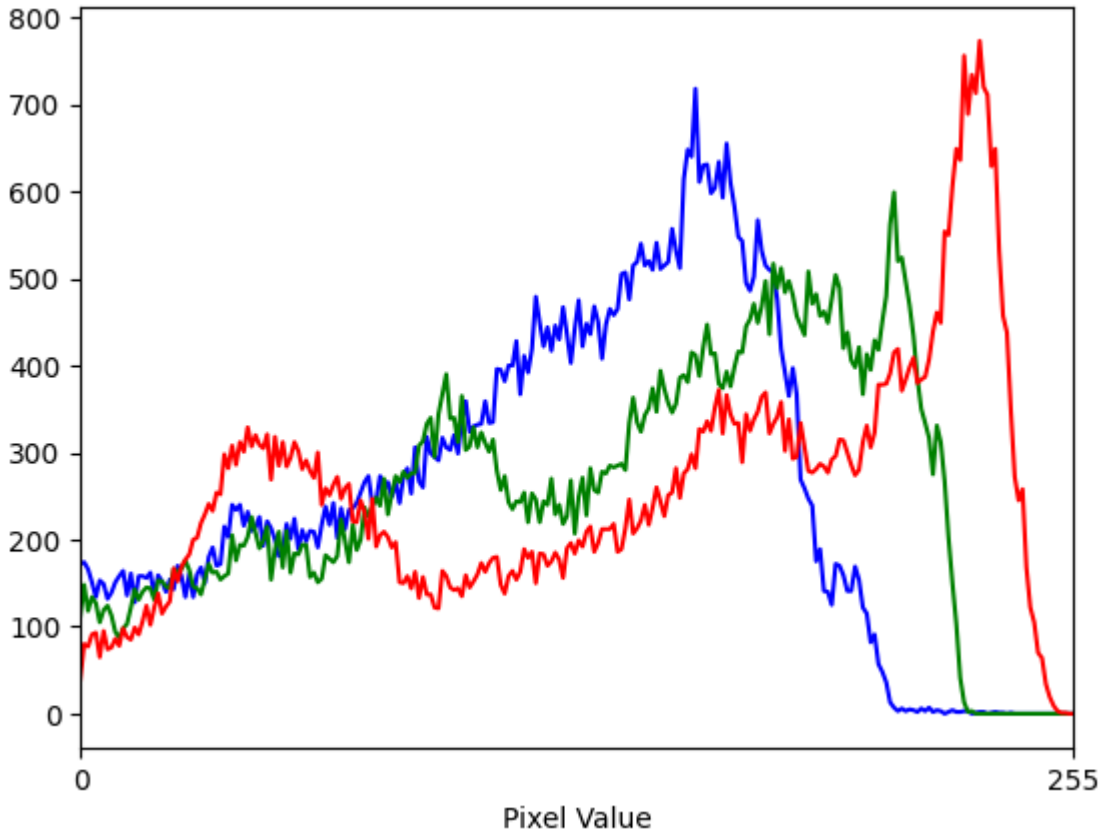
| 1990s- | 2000-2005 | 2006- |
|---|---|---|
| カラーヒストグラム | SIFT | Joint Haar-like |
| Wavelet | SURF | |
| LBP | Haar-like | Joint HOG |
| 高次局所自己相関 | HOG | |

- Joint Haar-likeはHaar-likeの組み合わせ
- Joint HOGはHOGの組み合わせ

# カラーヒストグラム

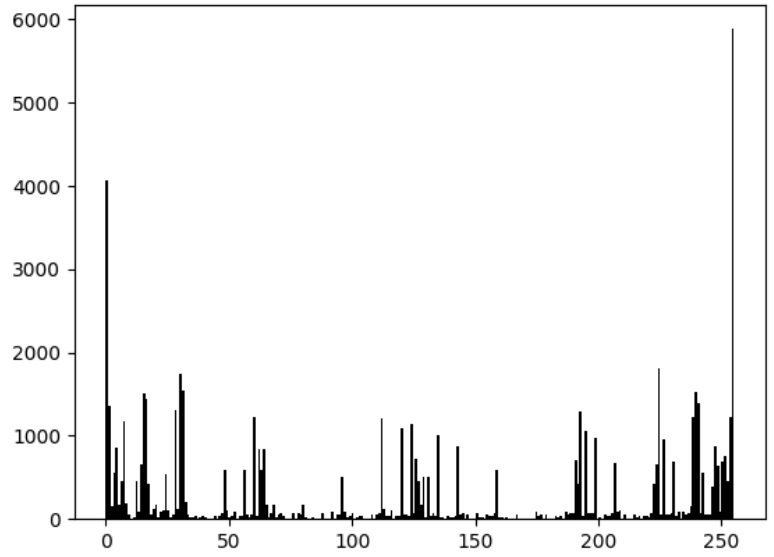
ヒストグラム≒256次元のベクトル

- 大域的特徴量のひとつ
- 縦軸は画素値の出現回数
- 横軸は画素値



# LBP

- **Local Binary Pattern**
- 局所的特徴量のひとつ
- 3×3の領域における中心画素と周辺画素の関係性を表現
- **Uniform LBP**はLBPを拡張した改良版



# Haar-like

- 局所特徴量のひとつ
- パターンを設定した矩形領域内の平均明度の差分を特徴量とする
- （黒の領域の平均値）ー（白の領域の平均値）
- 設定したパターンによって得られる特徴量が変化する

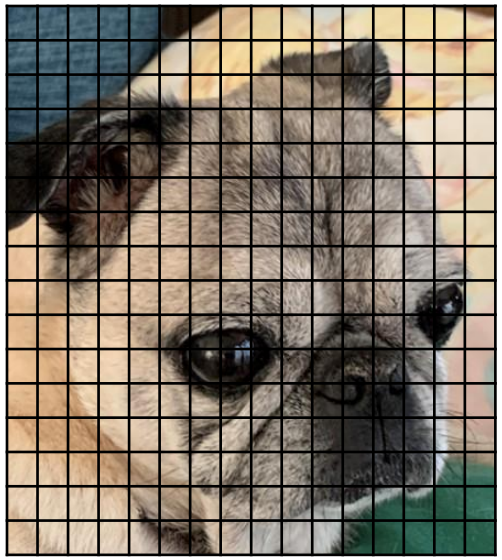
| | | |
|---|---|---|
| 30 | 24 | 56 |
| 122 | 90 | 89 |
| 155 | 221 | 201 |

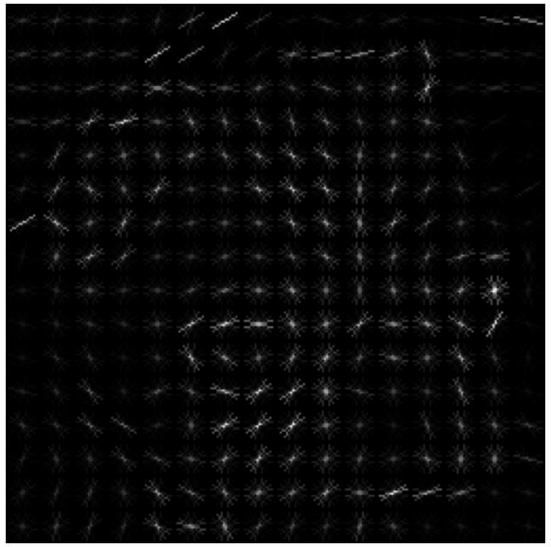
$$\frac{30+24}{2}-\frac{122+90}{2}=-79$$

| | |
|---|---|
| -79 | -49.5 |
| -82 | -121.5 |

# HOG

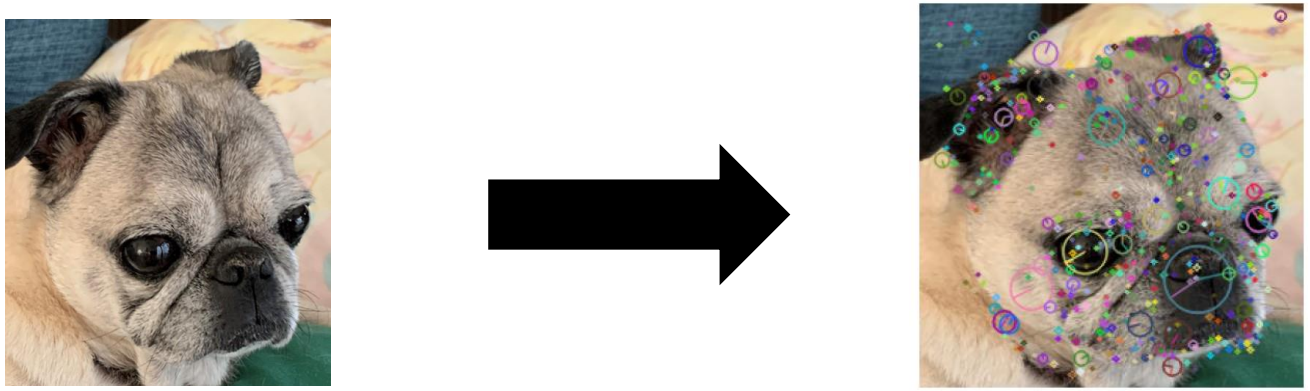
- **Histograms Of Gradients**
- 局所的特徴量のひとつ
- 局所領域（**セル**）における画素値の勾配方向をヒストグラムで表現
- セルの数×勾配方向ヒストグラムのビン数＝ベクトルの次元数





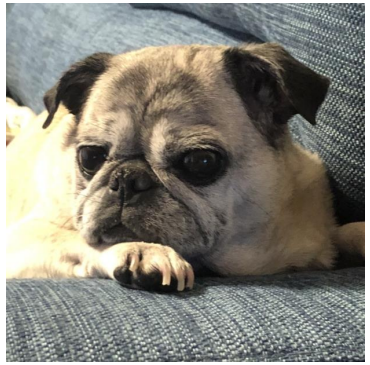
# SIFT

- **Scale Invariant Feature Transform**
- 局所的特徴量のひとつ
- 特徴点の検出と各特徴点の勾配方向ヒストグラムの計算を行う
- HOGはセル単位、SIFTは特徴点単位
- 特許取得技術（現在は期限切れ）
- **SURF（Speed-Up Robust Features）**はSIFTの処理速度改良版



# 従来の画像特徴量の限界

- 特徴量には不変性と弁別性が求められる
  敏感さと鈍感さ、局所性と大局性
- 人類は不変性と弁別性を併せ持つ万能な特徴量を考案できなかった
  もしかしたらこれから発見されるかもしれない...
- したがって従来の画像認識システムの精度は頭打ちとなっていた



精度低下要因の例

- 物体の角度や大きさが異なる
- 撮影状況が異なる
- 物体の一部が遮蔽物で隠れている
- ノイズが含まれている

# 7.3 畳み込みニューラルネットワーク

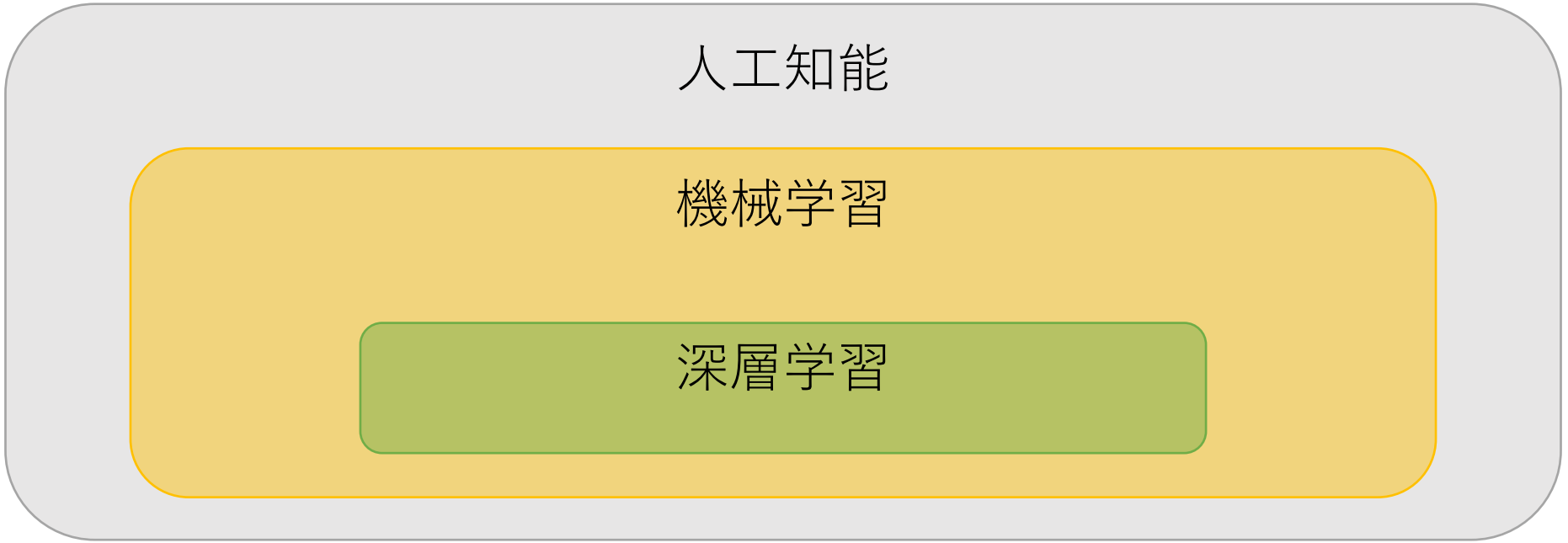
現代の画像認識AIの基礎となる超重要技術の仕組みとは

# 畳み込みニューラルネットワークとは

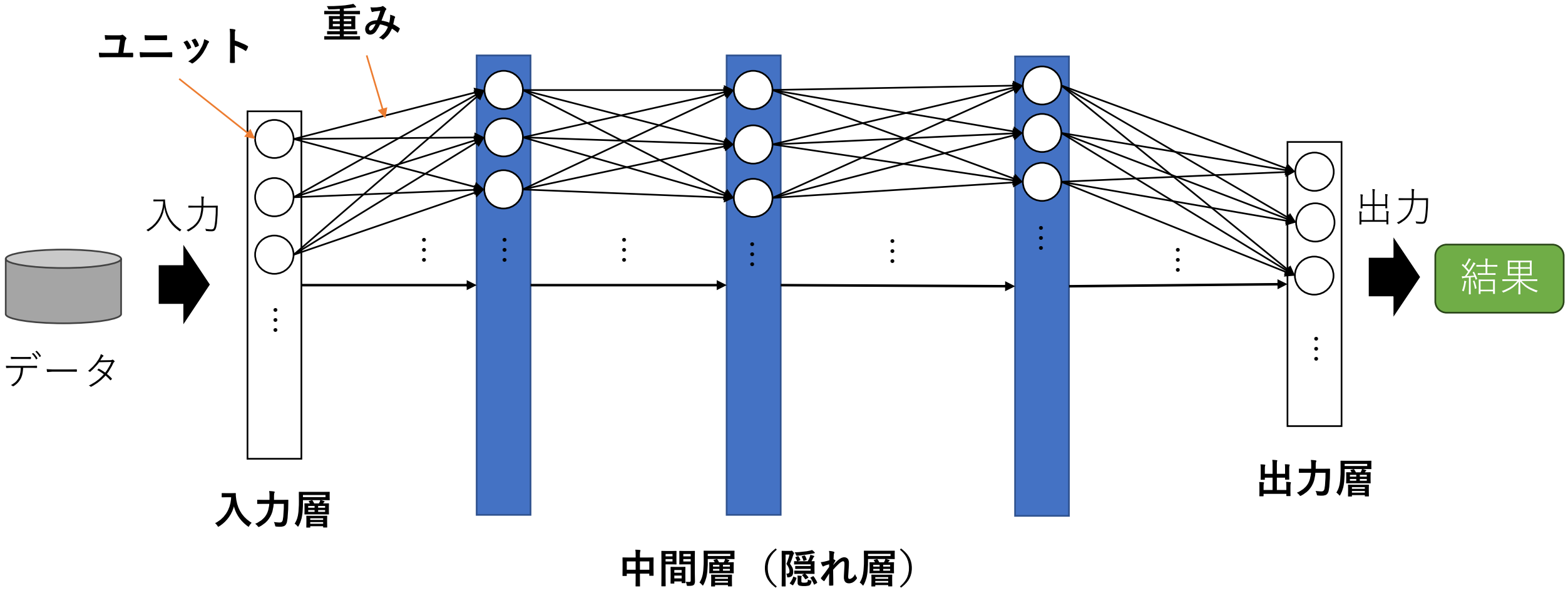
- **Convolutional Neural Network (CNN)**
- **ニューラルネットワーク（NN）**に**畳み込み処理**を導入したもの
- 画像に特化したネットワーク構造をもつ
- 1998年にYann LeCunが提案したLeNetが元祖CNN
  Yann LeCun：2019年にチューリング賞を受賞
- 現代の高精度な画像認識AIの多くはCNNを用いている

# （補足）深層学習とは

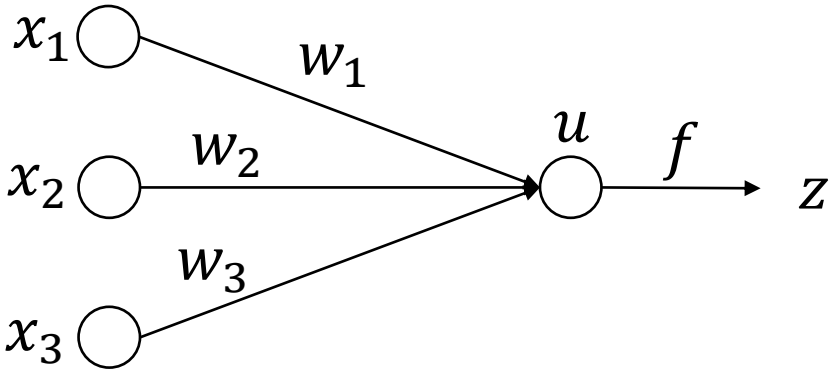
- 中間層が2層以上のNNを**Deep Neural Network (DNN)**という
- DNNの学習を**深層学習**という



# ニューラルネットワーク



# ユニットの計算処理




- 各ユニットは以下のような計算を行う

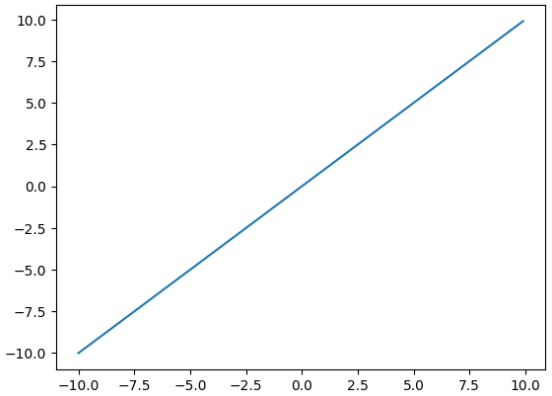
$$u = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$
$$z = f(u)$$

- $x_1, x_2, \ldots, x_n$を入力ベクトルとし、ユニットは$z$を出力する
- $w_1, w_2, \ldots, w_n$を**重み**、$b$を**バイアス**、$f$を**活性化関数**という
- 重みとバイアスは**パラメータ**（$\mathbf{w}$と表す）とも呼ばれる
- NNの学習＝最適なパラメータを求めること
- $z$は次の層の入力となる

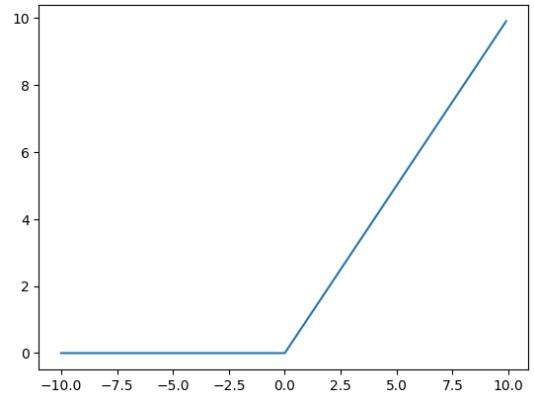
# 活性化関数

<u>よく使われる活性化関数</u>

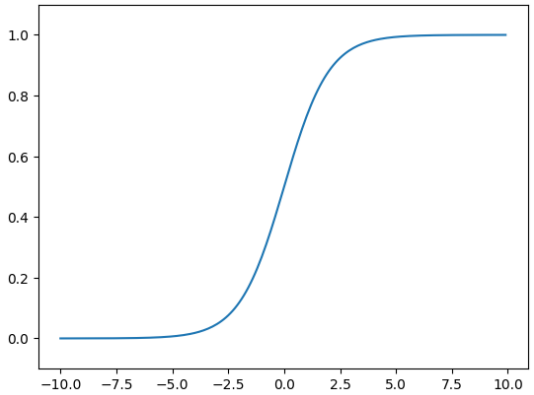
- 恒等関数（Linear）
  $f(x) = x$　つまりそのまま出力
  主に回帰の出力層で使用
- ReLU（Rectified Linear Unit）
  主に中間層で利用
- Sigmoid
  主に出力層で利用
- Softmax
  主に多クラス分類の出力層で利用



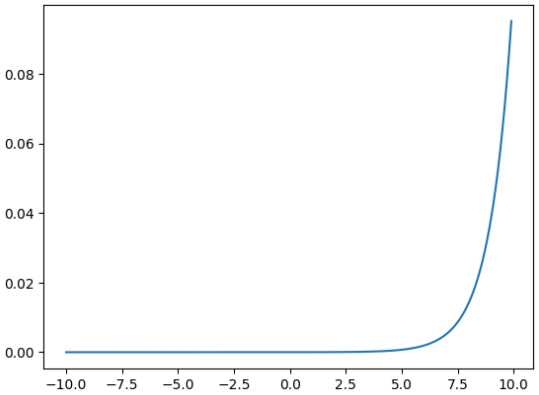

恒等関数




ReLU



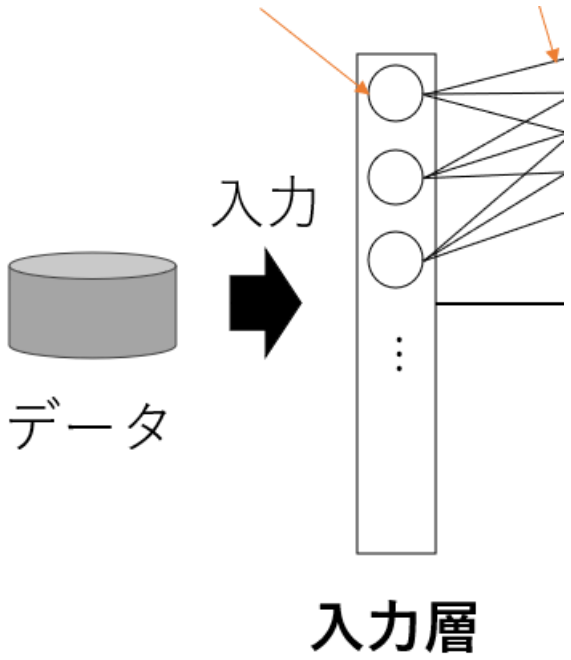

Sigmoid



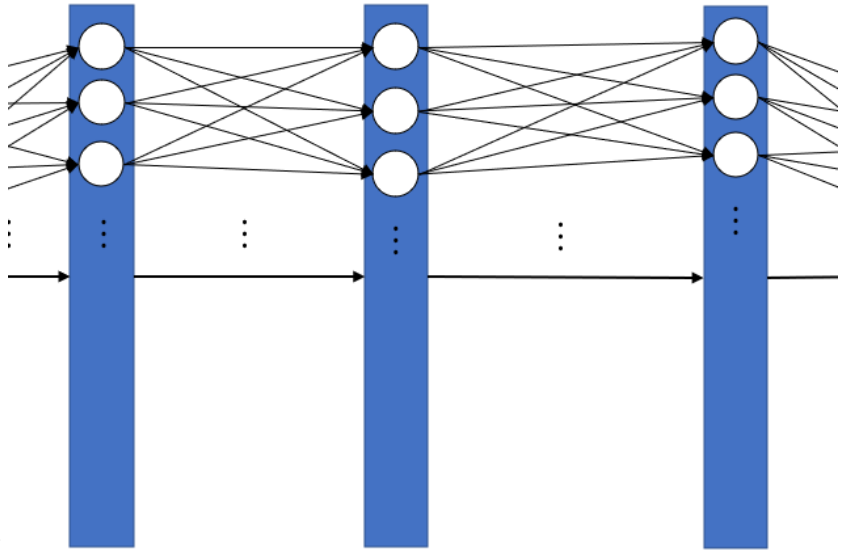

Softmax

# 入力層

- データを入力する層
- 入力層の次元はデータに合わせて適切に設計する
- 基本的にはデータに対して**正規化**や**標準化**を行ったあとに入力する
- 画像の場合は正規化を行うことが多い
  画素値を[0, 255]→[0, 1]に変換する



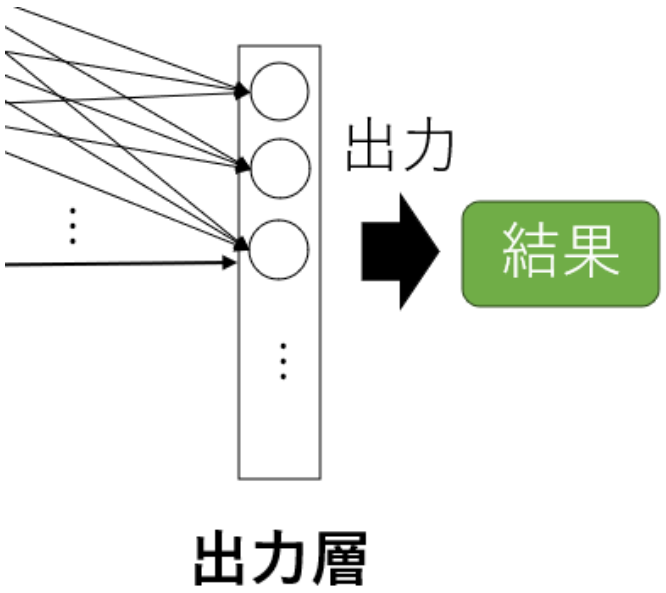
# 中間層




- さまざまな計算を行う層
- 各層におけるユニットの数は**任意**
- 層の数は**任意**
- 学習前のパラメータは適当な値で初期化する
  パラメータの初期化方法については割愛する
- 中間層を増やすとより複雑な表現を学習できるようになる

# 出力層

- 出力を得る層
- 各ユニットには目的の出力となるような活性化関数を適用する
  例）確率が欲しい→Sigmoid、数値が欲しい→恒等関数
- 画像認識ではSoftmaxやSigmoidが多い



# （余談）ニューラルネットワークの歴史

- 初期のNNが考案されたのは1980-1990sの第2次AIブームのとき
- 当時は中間層が数層程度の浅いモデルであった
- 現代では数百層程度の深いモデルも多数存在する

<u>現代になってAIが再燃した理由</u>

- デジタル化が進んだことで大規模なデータを収集できるようになった
- 計算機の性能が向上した

# ニューラルネットワークの教師あり学習

1. データをNNに入力してNNの出力を得る
2. 出力と教師データの**誤差（Loss）**を計算する
   - 誤差を計算する関数を**誤差関数（Loss Function）**という
   - 誤差関数は目的に合わせて適切に設計する必要がある
3. 誤差をもとにパラメータ**w**を更新する
   - 誤差を出力層から入力層方向に伝播させることを**誤差逆伝播**という
   - パラメータの更新方法については後述
4. 1-3を全データに対して実行する
5. 4を任意回数だけ繰り返す
   - この回数を**学習回数（エポック数）**という

# わかりやすさのために

これより先は以下の問題を想定する

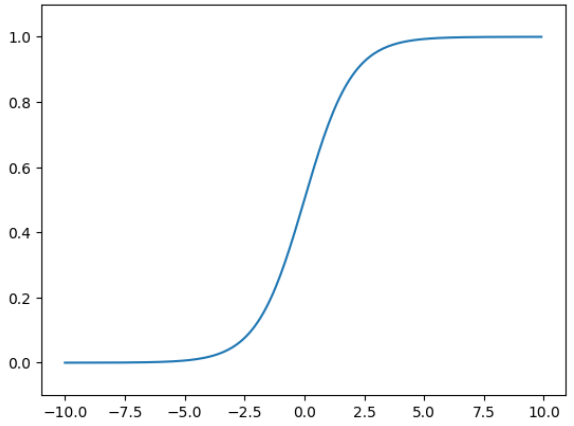
【問題】入力画像はパグですか？

# ニューラルネットワークの設計

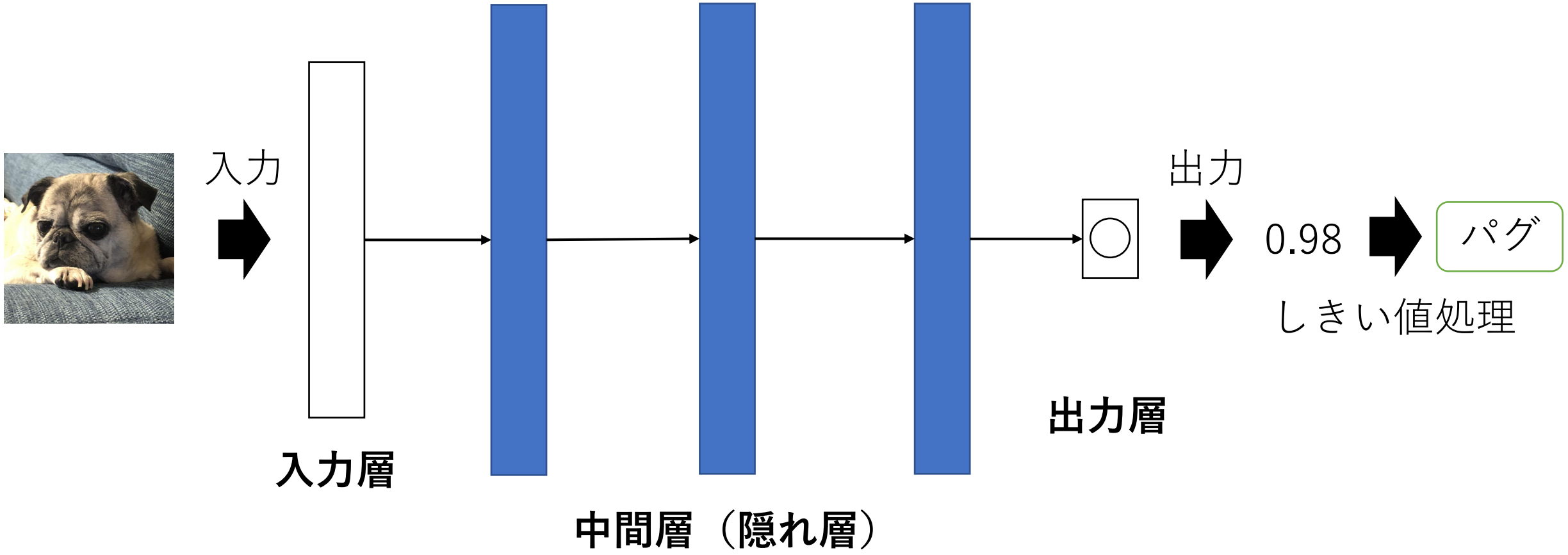
- 今回の問題はパグであるか否かの**二値分類**
- 出力はパグである確率が出てほしい

⬇

- 教師データはパグであれば1、そうでなければ0とする
- 出力層のユニット数は**1**とする
- 出力層の活性化関数は**Sigmoid**とする
  [0, 1]の値を出力したい



# 設計イメージ



# 誤差関数の設計

- 二値分類では**二値交差エントロピー（Binary Cross Entropy）**を用いる

$$E(\mathbf{w}) = -[t \log y(\mathbf{x}) + (1-t) \log\{1 - y(\mathbf{x})\}]$$

- 計算結果を$y(\mathbf{x})$、教師データを$t = \{0, 1\}$とする
- $t = 1$のときは$-\log y(\mathbf{x})$、$t = 0$のときは$-\log\{1 - y(\mathbf{x})\}$が誤差となる

  例）計算結果が0.78、正解がパグである（$t = 1$）のとき

  $E = -\log 0.78 = -(-0.1079) = 0.1079$

# パラメータの更新

- 以下の式によるパラメータの更新方法を**勾配降下法** という

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$

- $\alpha$は**学習率**といい、パラメータの更新度合いを決める値
  学習率は人があらかじめ決めておく
- 誤差を偏微分した値（勾配）を用いてパラメータを更新する
- 勾配降下法にはさまざまな進化系がある
  モメンタム、RMSProp、Adamなど
- 最近では「とりあえずAdam」というイメージ

# 勾配降下法のイメージ

- 勾配が+なら-方向にパラメータを更新
- 勾配が-なら+方向にパラメータを更新
- 最も誤差が小さい点への収束を目指す

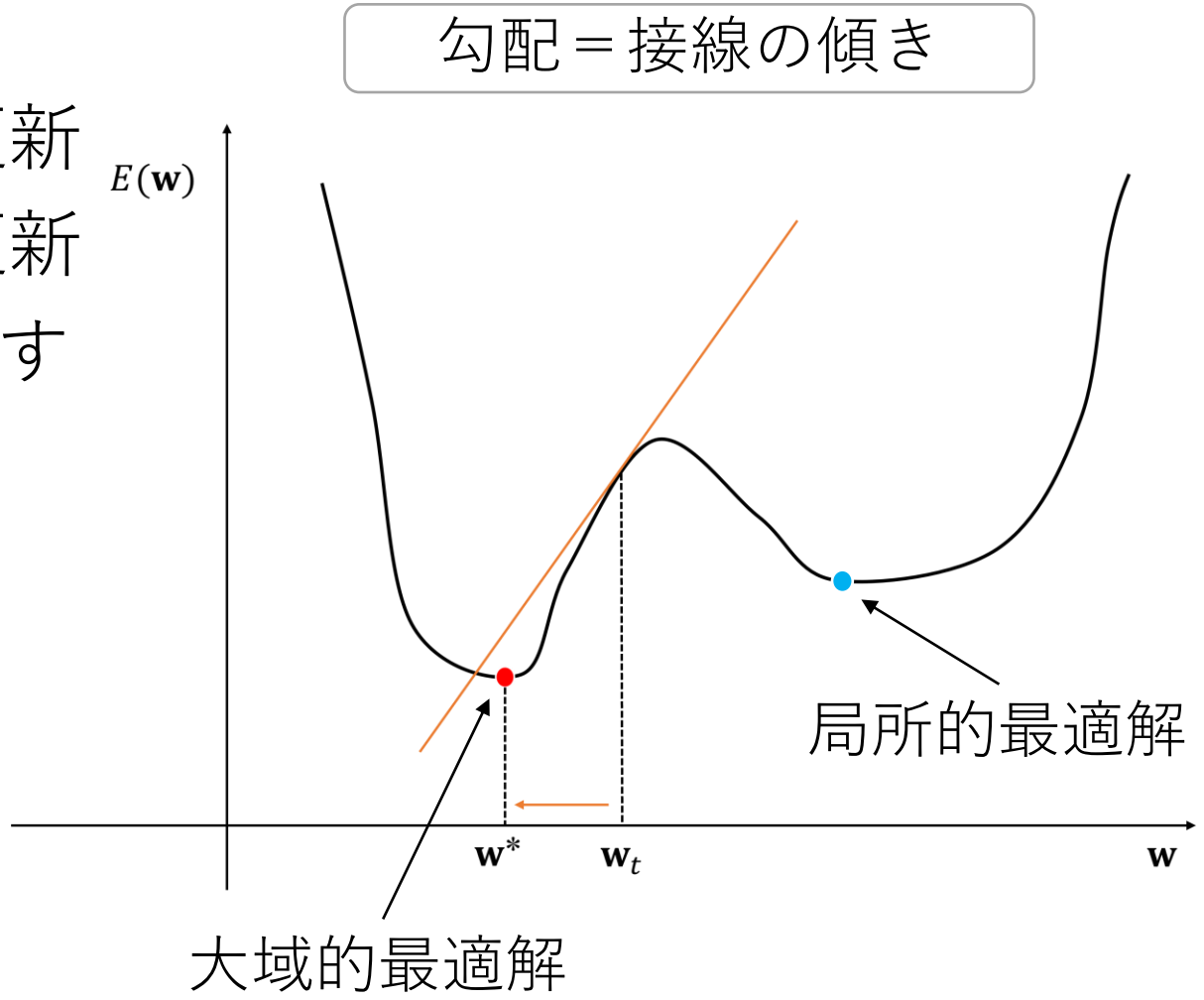
<u>学習率の影響</u>

学習率大

- 局所的最適解に収束する可能性が低い
- 収束が遅い

学習率小

- 局所的最適解に収束する可能性が高い



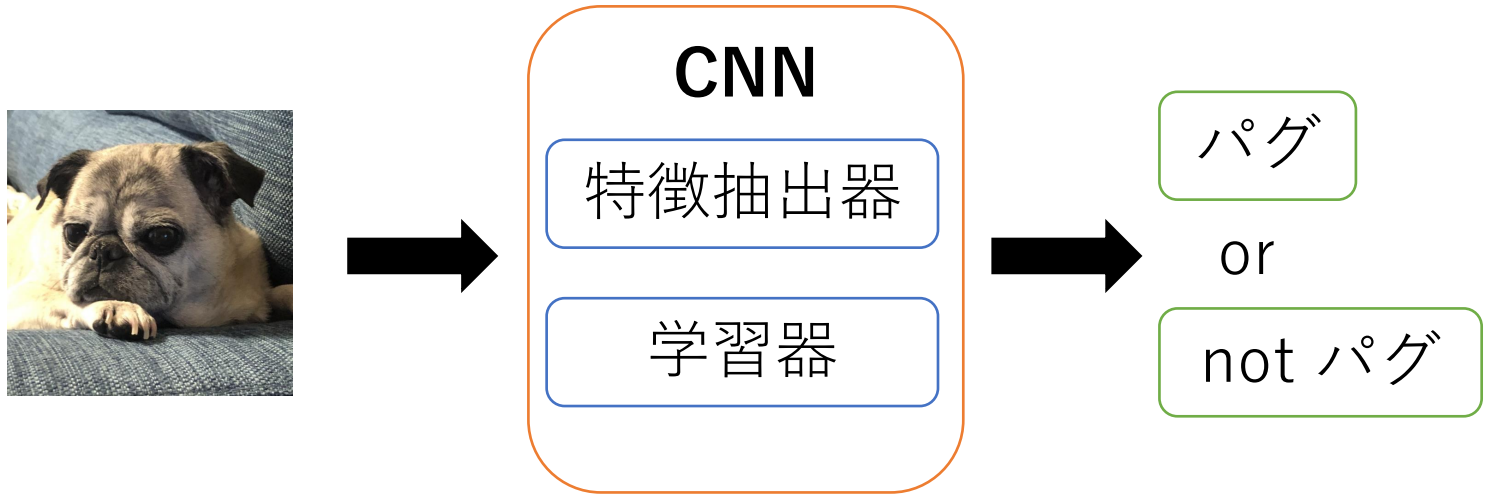
# 学習方法の種類

| | バッチ学習 | オンライン学習 | ミニバッチ学習 |
|---|---|---|---|
| 1回の学習に用いる誤差 | 全データの平均誤差 | 1つのデータの誤差 | グループの平均誤差 |
| 学習の速さ | 高速 | 低速 | バランス |
| 局所的最適解の危険性 | 高い | 低い | バランス |

- ミニバッチ学習におけるグループを**ミニバッチ**という
- ミニバッチに含まれるデータ数を**バッチサイズ**という
  例）バッチサイズが16→16個のデータの平均誤差を学習に用いる
- 一般的にはミニバッチ学習が主流

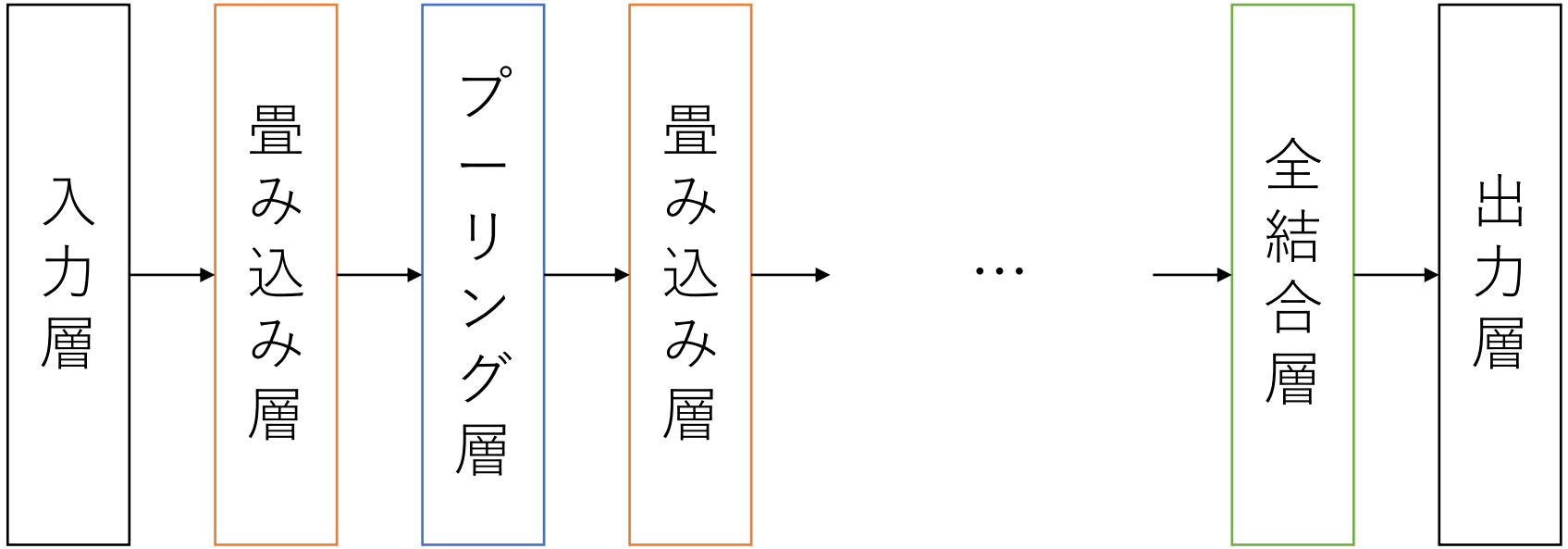
# 畳み込みニューラルネットワークの構造

- 主に**畳み込み層**、**プーリング層**、**全結合層**で構成される
- 畳み込み層とプーリング層は特徴量抽出の役割をもつ
- 全結合層は学習器の役割をもつ
- CNNは一人二役の**エンドツーエンド**な手法



# 畳み込みニューラルネットワークの特徴

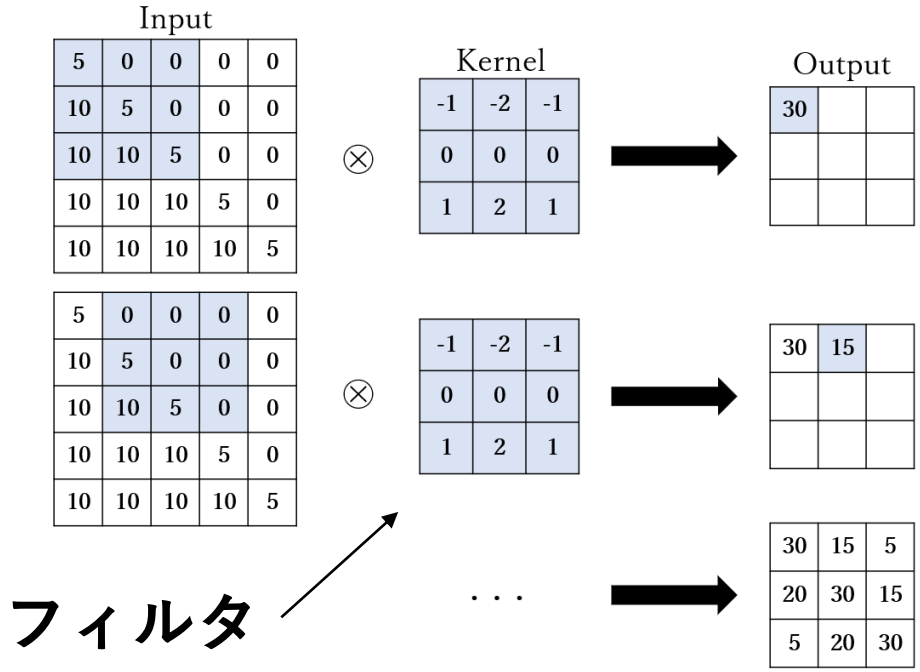
- 畳み込み層とプーリング層を何度か繰り返して全結合層につながる
- プーリング層により特徴の位置ずれを吸収
- 浅い層では局所的な、深い層では大局的な特徴量を抽出する



# 畳み込み層

$$u = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$
$$z = f(u)$$

- 畳み込み処理を行う層
- **フィルタ（カーネル）**を用いて**特徴マップ**を出力する
- フィルタの値はパラメータに相当するため、学習によって決定される




この例では...
**ストライド**　1
**フィルタサイズ**　3×3

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

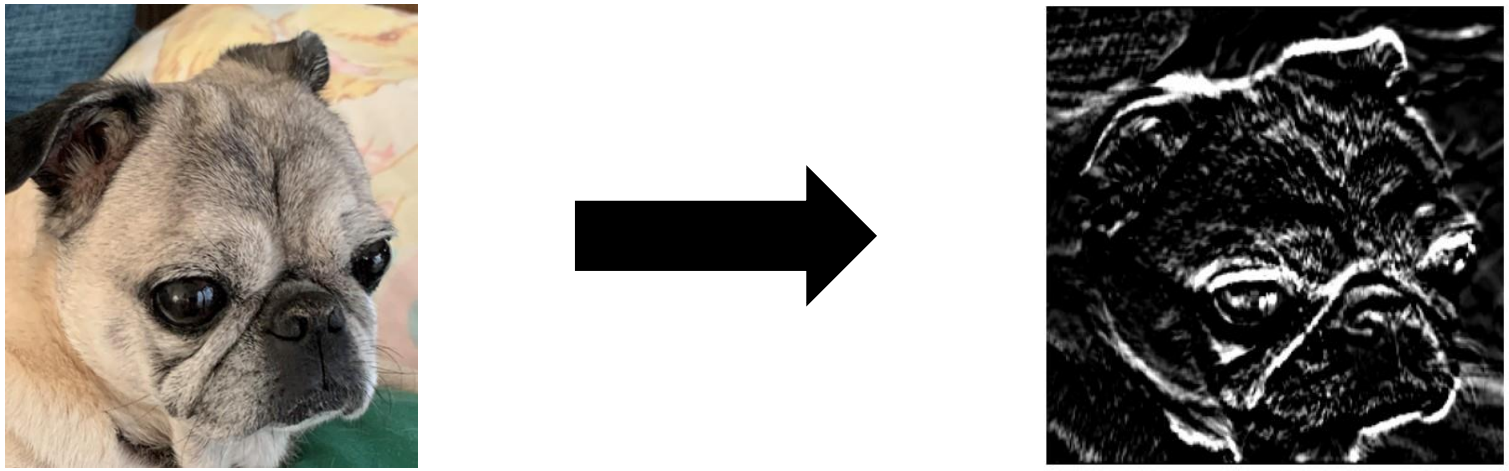
フィルタ

# （補足）画像のフィルタ処理

| 1 | 2 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Sobelフィルタ

- フィルタ処理＝畳み込み処理
- 以下は縦方向Sobelフィルタのフィルタ処理結果



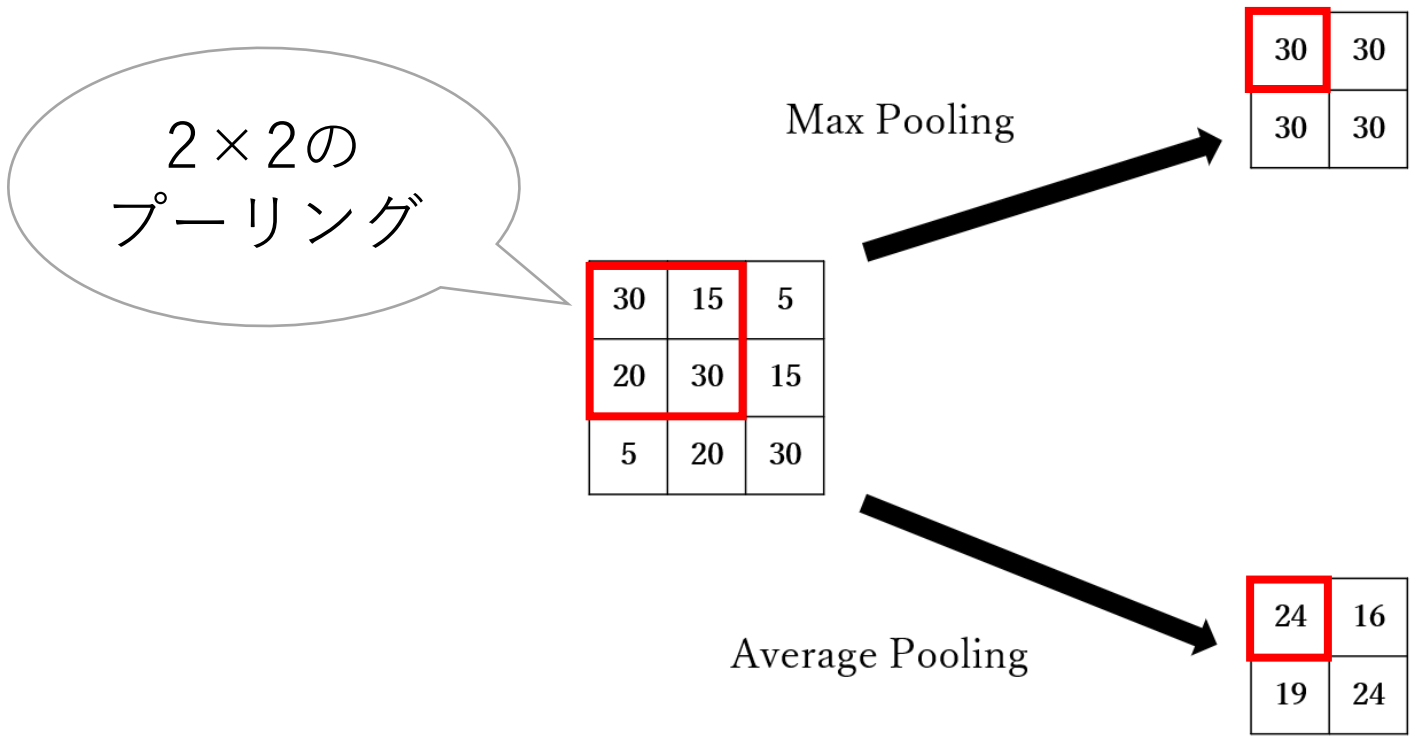
- CNNではこのフィルタの値を学習によって自動的に決定する

# プーリング層

- 畳み込み層の出力を集約する処理（**プーリング処理**）を行う層
- プーリング処理により特徴の微小な位置変化に対して頑健になる

<u>プーリングの種類</u>

- 最大値プーリング
  範囲内の最大値をとる
- 平均値プーリング
  範囲内の平均値をとる



# 全結合層

- 前の層の出力を任意長の1次元ベクトルにまとめる層
- 一般的なCNNでは出力層の前に設置する
  複数設置することもある
- 計算処理はNNと同じ

$$u = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$
$$z = f(u)$$

一般的なCNNにおける実際の処理

1. 最後のプーリング層の出力を1次元ベクトルに平坦化する
2. ベクトルの全要素と全結合層を重み付きで結合する
   この計算はNNのユニットの計算と同じ
3. 全結合層と出力層も同様に結合する

# まとめ

- 従来の画像認識についておさらいした
- 主要な画像特徴量についておさらいした
- CNNの基礎を学んだ

【次回】
- CNNを実装する

# 演習

- OpenCVとScikit Imageを用いて画像特徴量を抽出する
- 活性化関数を実装して動きを見る
- 畳み込み処理を実装して動きをみる

サンプルプログラム

https://github.com/wt501/sample_programs